f# for data analysis

Артём Акуляков

Я

- dotnet, python, js, go
- *f*(λ), data analysis, ml
- fintech стартап, senior engineer

data analysis

data analysis

- math and computer science
- состоит
    - очистка
    - трансформация
    - дополнение
    - фильтрация
    - моделирование
    - кластеризация
    - поиск корреляций
    - проверка гипотез
    - ...
- … rocket sience

data analysis не всегда rocket sience

- проект на поддержку без описания структуры данных
- расследование по логам
- сложная аналитика
- ...

житейский
data analysis

f#

f# разработчики самые счастливые

f#

```
let rec map func lst =
    match lst with
        | [] -> []
        | head :: tail -> func head :: map func tail

let myList = [1;3;5]
let newList = map (fun x -> x + 1) myList
```

f#

- functional-first programming language
- компилируемый & интерпретируемый
- dotnet
- linux, osx, win, +
- ~~монады, матан и вся страшная жесть~~
- хорош для data analysis

data analysis?

data analysis

1. доступ к данным
2. визуализация
3. манипулирование данными
4. интеграция(?)

1 доступ к данным

1 доступ к данным

FSharp.Data & FSharp.Data.TypeProviders

- sql db
- web & files
  - json
  - xml
  - csv
  - html
- world bank
- twitter
- ...

FSharp.Data

```
open FSharp.Data

type GitHub = JsonProvider<"https://api.github.com/repos/dotnet/coreclr/issues">

let topRecentlyUpdatedIssues =
   GitHub.GetSamples()
   |> Seq.sortBy (fun x -> x.UpdatedAt)
   |> Seq.truncate 5
```

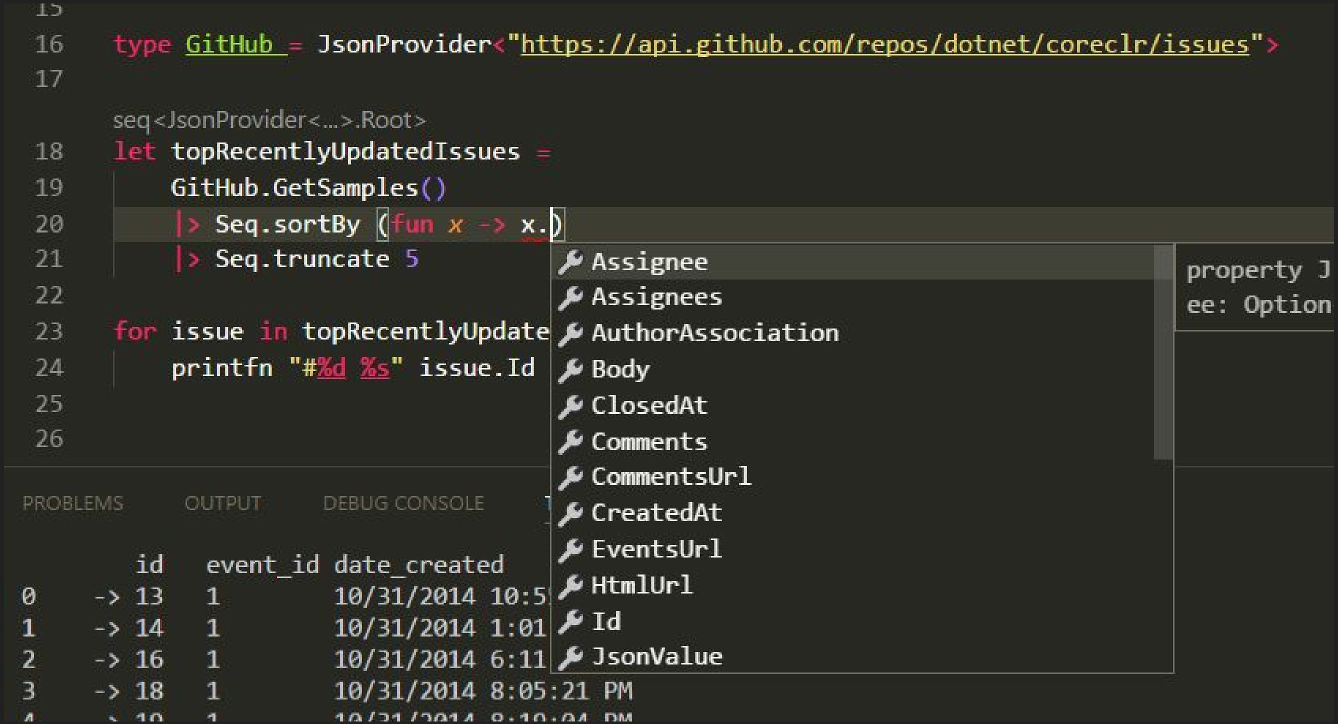
FSharp.Data

```fsharp
open FSharp.Data

type GitHub = JsonProvider<"https://api.github.com/repos/dotnet/coreclr/issues">

let topRecentlyUpdatedIssues =
    GitHub.GetSamples()
    |> Seq.sortBy (fun x -> x.UpdatedAt)
    |> Seq.truncate 5
```

FSharp.Data

```
open FSharp.Data

type GitHub = JsonProvider<"https://api.github.com/repos/dotnet/coreclr/issues">

let topRecentlyUpdatedIssues =
    GitHub.GetSamples()
    |> Seq.sortBy (fun x -> x.UpdatedAt)
    |> Seq.truncate 5
```

FSharp.Data



```
type GitHub = JsonProvider<"https://api.github.com/repos/dotnet/coreclr/issues">

seq<JsonProvider<...>.Root>
let topRecentlyUpdatedIssues =
    GitHub.GetSamples()
    |> Seq.sortBy (fun x -> x.)
    |> Seq.truncate 5

for issue in topRecentlyUpdate
    printfn "#%d %s" issue.Id
```

```
      id   event_id date_created
0  -> 13   1        10/31/2014 10:5
1  -> 14   1        10/31/2014 1:01
2  -> 16   1        10/31/2014 6:11
3  -> 18   1        10/31/2014 8:05:21 PM
```

2 визуализация

2 визуализация

- xplot
- FSharp.Charting

FSharp.Charting

```
open FSharp.Charting
open System

let d1 = [for x in 0 .. 100 -> (x, 1.0 / (float x + 1.) )]
let d2 = [for x in 0 .. 100 -> (x, Math.Sin(float(x)))]

Chart.Rows [
   Chart.Line(d1,Name="d1",Title="d1")
   Chart.Column(d2,Name="d2",Title="d2")
]
```

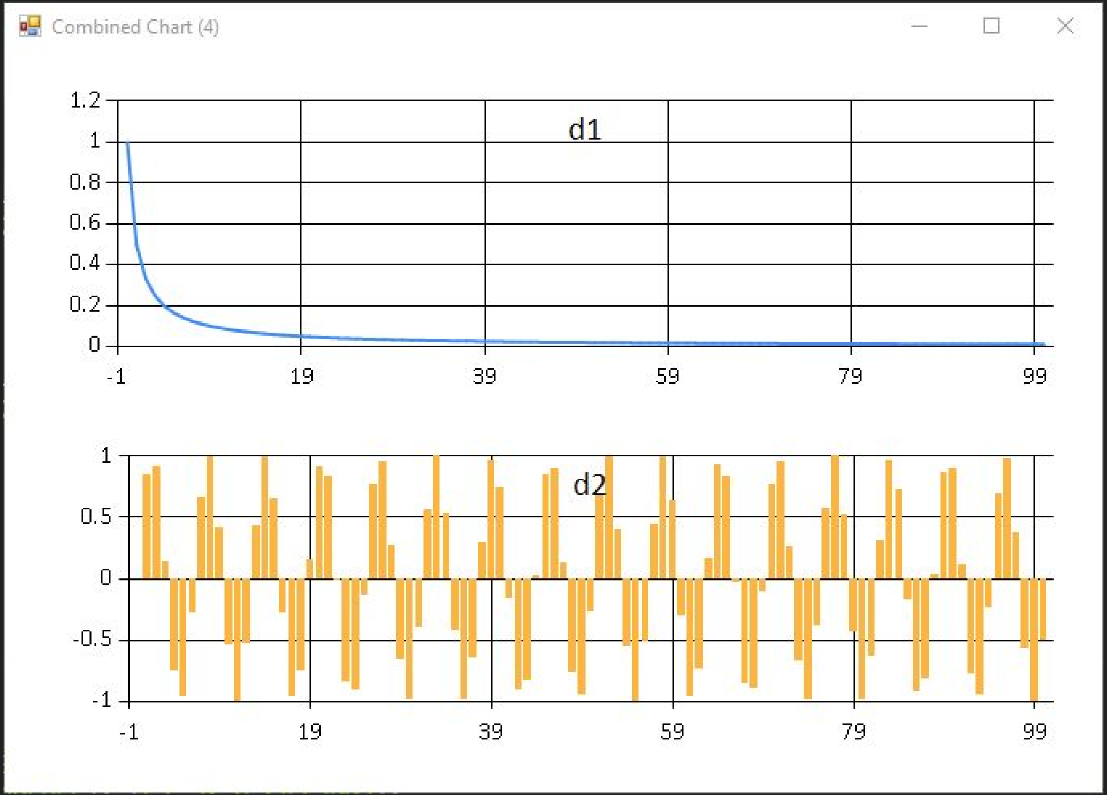
FSharp.Charting

```
open FSharp.Charting
open System

let d1 = [for x in 0 .. 100 -> (x, 1.0 / (float x + 1.) )]
let d2 = [for x in 0 .. 100 -> (x, Math.Sin(float(x)))]

Chart.Rows [
   Chart.Line(d1,Name="d1",Title="d1")
   Chart.Column(d2,Name="d2",Title="d2")
]
```

FSharp.Charting

```
open FSharp.Charting
open System

let d1 = [for x in 0 .. 100 -> (x, 1.0 / (float x + 1.) )]
let d2 = [for x in 0 .. 100 -> (x, Math.Sin(float(x)))]

Chart.Rows  [
   Chart.Line(d1,Name="d1",Title="d1")
   Chart.Column(d2,Name="d2",Title="d2")
]
```

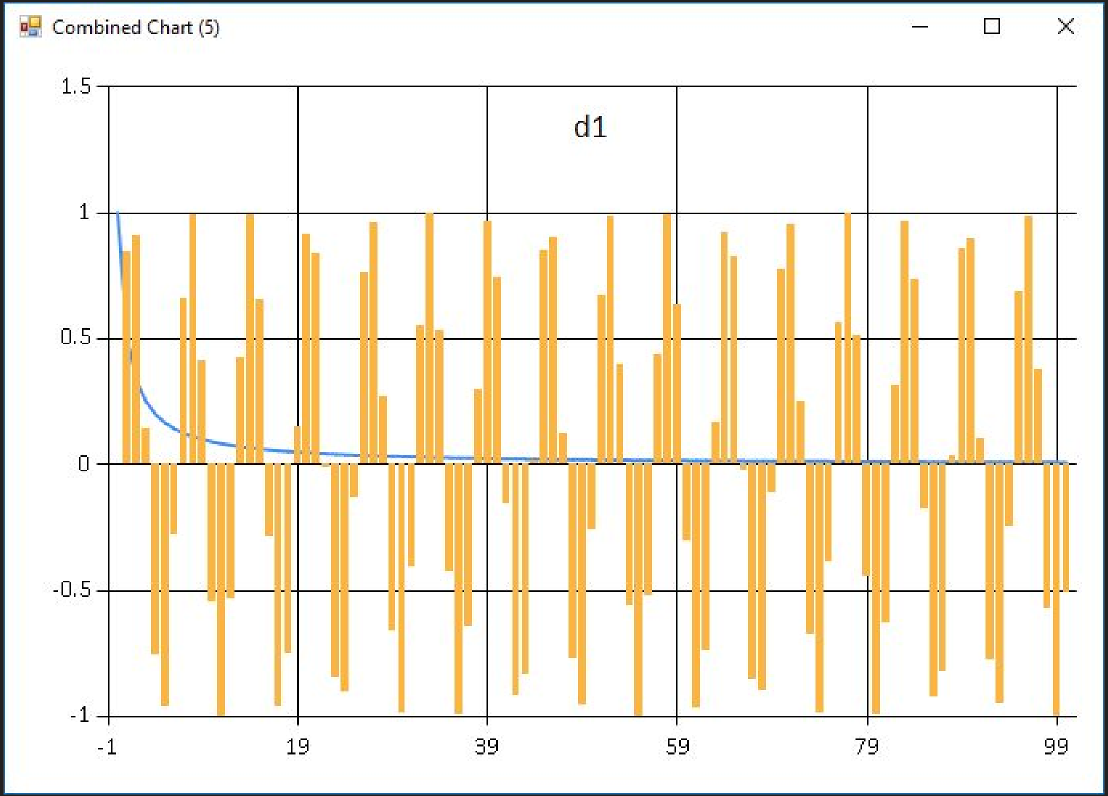
FSharp.Charting



FSharp.Charting

```
open FSharp.Charting
open System

let d1 = [for x in 0 .. 100 -> (x, 1.0 / (float x + 1.) )]
let d2 = [for x in 0 .. 100 -> (x, Math.Sin(float(x)))]

Chart.Combine [
   Chart.Line(d1,Name="d1",Title="d1")
   Chart.Column(d2,Name="d2",Title="d2")
]
```

FSharp.Charting



3 манипулирование данными

3 манипулирование данными

- repl & $f(\lambda)$

3 манипулирование данными

- repl & $f(\lambda)$
- Deedle

Deedle

- аналог pandas из мира python
- недо-Excel на стероидах
- приправа из статистики

Deedle

- Series
- Frames
- Stats

Deedle

```fsharp
open FSharp.Data

let WorldBank = WorldBankData.GetDataContext()

let co2Indicator =
    WorldBank
        .Countries.``Russian Federation``
        .Indicators.``CO2 emissions (metric tons per capita)``

let populationIndicator =
    WorldBank
        .Countries.``Russian Federation``
        .Indicators.``Population, total``
```

Deedle

```
open Deedle
open FSharp.Data

let WorldBank = ...

let co2Series = co2Indicator |> Series.ofObservations
let populationSeries = populationIndicator |> Series.ofObservations

let frame = Frame(["co2";"population"],[co2Series;populationSeries])|>Frame.dropSparseRows

frame?totalCo2 <- frame?co2 * frame?population

let avarageCo2 = frame?totalCo2 |> Stats.median
```

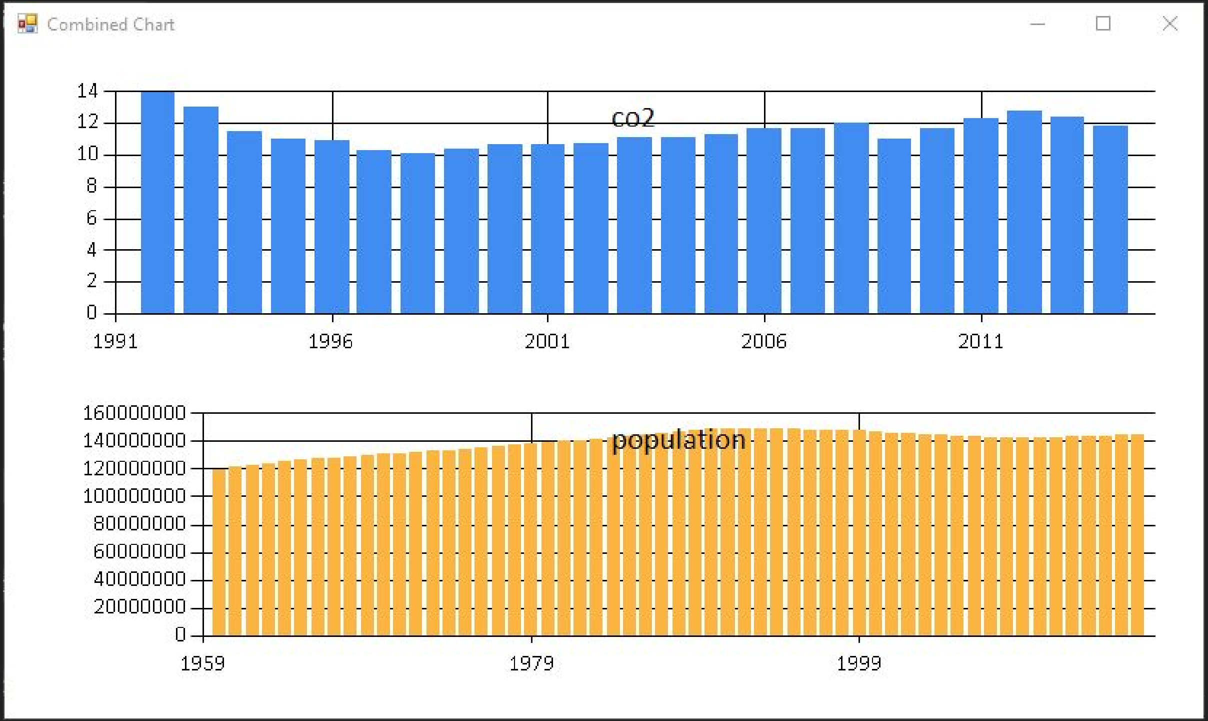
Deedle

```fsharp
open Deedle
open FSharp.Data

let WorldBank = ...

let co2Series = co2Indicator |> Series.ofObservations
let populationSeries = populationIndicator |> Series.ofObservations

let frame = Frame(["co2";"population"],[co2Series;populationSeries])|>Frame.dropSparseRows

frame?totalCo2 <- frame?co2 * frame?population

let avarageCo2 = frame?totalCo2 |> Stats.median
```

Deedle

```
open Deedle
open FSharp.Data

let WorldBank = ...

let co2Series = co2Indicator |> Series.ofObservations
let populationSeries = populationIndicator |> Series.ofObservations

let frame = Frame(["co2";"population"],[co2Series;populationSeries])|>Frame.dropSparseRows

frame?totalCo2 <- frame?co2 * frame?population

let avarageCo2 = frame?totalCo2 |> Stats.median
```

Deedle



Combined Chart
co2
14
12
10
8
6
4
2
0
1991
1996
2001
2006
2011
population
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
1959
1979
1999

Deedle

```
open Deedle
open FSharp.Data

let WorldBank = ...

let co2Series = co2Indicator |> Series.ofObservations
let populationSeries = populationIndicator |> Series.ofObservations

let frame = Frame(["co2";"population"],[co2Series;populationSeries])|>Frame.dropSparseRows

frame?totalCo2 <- frame?co2 * frame?population

let avarageCo2 = frame?totalCo2 |> Stats.median
```

Deedle

```
open Deedle
open FSharp.Data

let WorldBank = ...

let co2Series = co2Indicator |> Series.ofObservations
let populationSeries = populationIndicator |> Series.ofObservations

let frame = Frame(["co2";"population"],[co2Series;populationSeries])|>Frame.dropSparseRows

frame?totalCo2 <- frame?co2 * frame?population

let avarageCo2 = frame?totalCo2 |> Stats.median
```

Deedle

```fsharp
open Deedle
open FSharp.Data

let WorldBank = ...

let co2Series = co2Indicator |> Series.ofObservations
let populationSeries = populationIndicator |> Series.ofObservations

let frame = Frame(["co2";"population"],[co2Series;populationSeries])|>Frame.dropSparseRows

frame?totalCo2 <- frame?co2 * frame?population

let avarageCo2 = frame?totalCo2 |> Stats.median
```

3 манипулирование данными

- repl & $f(\lambda)$
- Deedle
- Math.Net

Math.Net

- матричные вычисления
- статистика
- решение систем уравнений
- регрессия
- ...

Math.Net

```
open MathNet.Numerics.Statistics

...

let co2Values = frame?co2 |> Series.values
let populationValues = frame?population |> Series.values

let coef = Correlation.Spearman(populationValues, co2Values)
```

Math.Net

```
open MathNet.Numerics.Statistics

...

let co2Values = frame?co2 |> Series.values
let populationValues = frame?population |> Series.values

let coef = Correlation.Spearman(populationValues, co2Values)
```

Math.Net

```
open MathNet.Numerics.Statistics

...

let co2Values = frame?co2 |> Series.values
let populationValues = frame?population |> Series.values

let coef = Correlation.Spearman(populationValues, co2Values)
```

Math.Net

```
open MathNet.Numerics.Statistics

...

let co2Values = frame?co2 |> Series.values
let populationValues = frame?population |> Series.values

let coef = Correlation.Spearman(populationValues, co2Values)
?> -0.2756916996
```

3 манипулирование данными

- repl & $f(\lambda)$
- Deedle
- Math.Net
- R Provider

R Provider

- удобно получать и обрабатывать данные с f#
- удобно делать расчеты и визуализацию с R

R Provider

```
...
open RDotNet
open RProvider

let rng = Random()
let rand () = rng.NextDouble()

let X1s = [ for i in 0 .. 9 -> 10. * rand () ]
let X2s = [ for i in 0 .. 9 -> 5. * rand () ]
let Ys = [ for i in 0 .. 9 -> 5. + 3. * X1s.[i] - 2. * X2s.[i] + rand () ]

R.plot(Ys)
```

R Provider

```
...
open RDotNet
open RProvider

let rng = Random()
let rand () = rng.NextDouble()

let X1s = [ for i in 0 .. 9 -> 10. * rand () ]
let X2s = [ for i in 0 .. 9 -> 5. * rand () ]
let Ys = [ for i in 0 .. 9 -> 5. + 3. * X1s.[i] - 2. * X2s.[i] + rand () ]

R.plot(Ys)
```

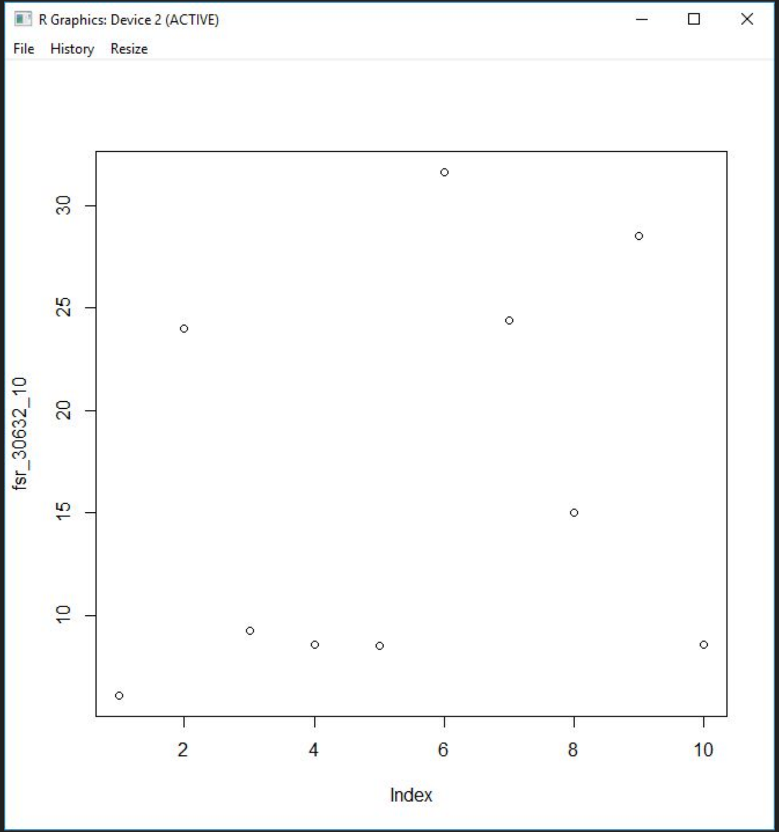
R Provider

```
...
open RDotNet
open RProvider

let rng = Random()
let rand () = rng.NextDouble()

let X1s = [ for i in 0 .. 9 -> 10. * rand () ]
let X2s = [ for i in 0 .. 9 -> 5. * rand () ]
let Ys = [ for i in 0 .. 9 -> 5. + 3. * X1s.[i] - 2. * X2s.[i] + rand () ]

R.plot(Ys)
```

R Provider

```
...
open RDotNet
open RProvider

let rng = Random()
let rand () = rng.NextDouble()

let X1s = [ for i in 0 .. 9 -> 10. * rand () ]
let X2s = [ for i in 0 .. 9 -> 5. * rand () ]
let Ys = [ for i in 0 .. 9 -> 5. + 3. * X1s.[i] - 2. * X2s.[i] + rand () ]

R.plot(Ys)
```

R Provider


R Graphics: Device 2 (ACTIVE)
File
History
Resize
fsr_30632_10
Index
10
15
20
25
30
2
4
6
8
10

R Provider

```
...

let ds = namedParams ["Y", box Ys;"X1", box X1s;"X2", box X2s;] |> R.data_frame
let result = R.lm(formula = "Y~X1+X2", data = ds)
```

R Provider

```
...

let ds = namedParams ["Y", box Ys;"X1", box X1s;"X2", box X2s;] |> R.data_frame
let result = R.lm(formula = "Y~X1+X2", data = ds)
```

R Provider

```
...

let ds = namedParams ["Y", box Ys;"X1", box X1s;"X2", box X2s;] |> R.data_frame
let result = R.lm(formula = "Y~X1+X2", data = ds)
```

R Provider

```
...

let ds = namedParams ["Y", box Ys;"X1", box X1s;"X2", box X2s;] |> R.data_frame
let result = R.lm(formula = "Y~X1+X2", data = ds)

?> Coefficients:
(Intercept)           X1           X2
      5.444        2.988       -1.926
```

R Provider

```
...

let Ys = [ for i in 0 .. 9 -> 5. + 3. * X1s.[i] - 2. * X2s.[i] + rand () ]

...

?> Coefficients:
(Intercept)           X1           X2
      5.444        2.988       -1.926
```

3 манипулирование данными

- repl & $f(\lambda)$
- Deedle
- Math.Net
- R Provider
- {m}brace

{m}brace

- azure
- aws(?)

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

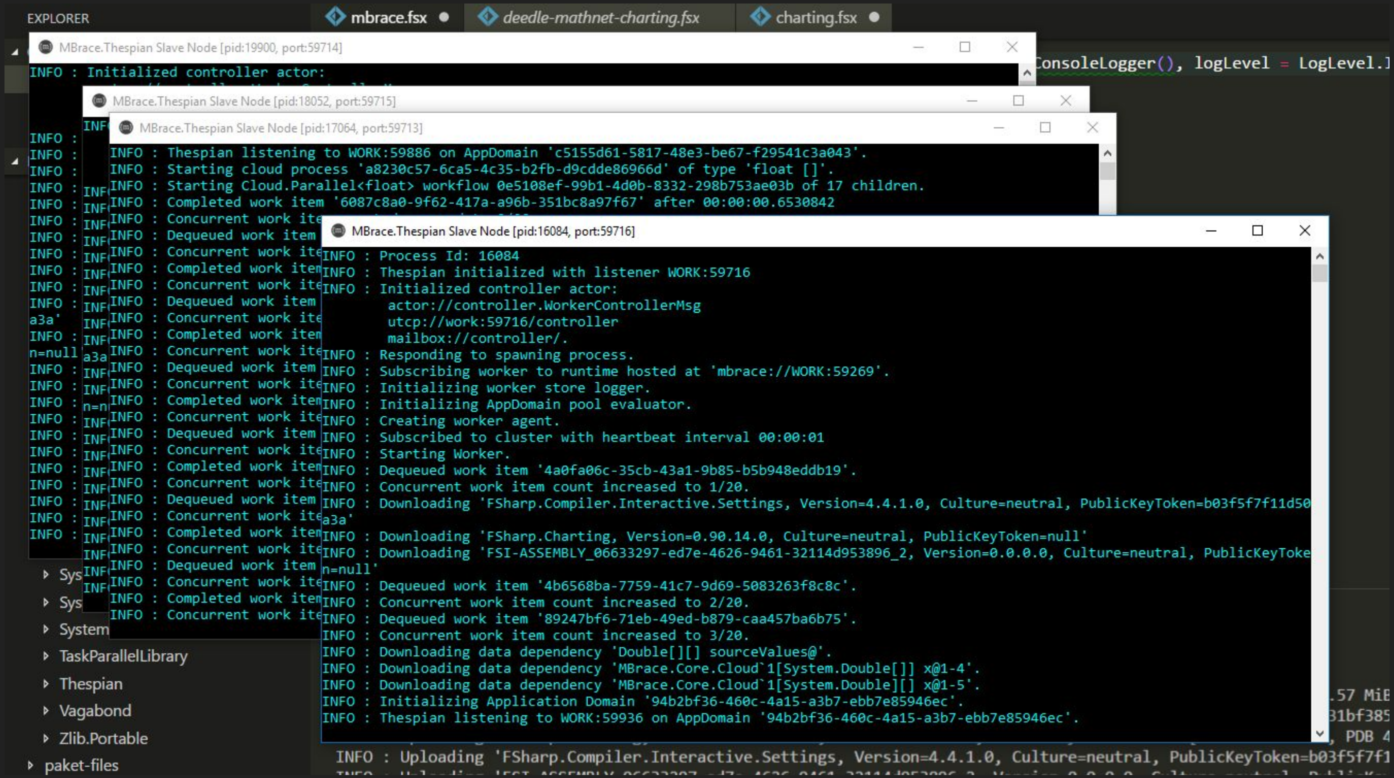
{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace



```
onsoleLogger(), logLevel = LogLevel.I
```

MBrace.Thespian Slave Node [pid:19900, port:59714]

```
INFO : Initialized controller actor:

INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
a3a'
INFO :
n=null
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
INFO :
```

MBrace.Thespian Slave Node [pid:18052, port:59715]

```
INF

INF
INF
INF
INF
INF
INF
INF
INF
INF
INF
a3a
INF
INF
n=n
INF
INF
INF
INF
INF
INF
INF
INF
INF
INF
INF
```

MBrace.Thespian Slave Node [pid:17064, port:59713]

```
INFO : Thespian listening to WORK:59886 on AppDomain 'c5155d61-5817-48e3-be67-f29541c3a043'.
INFO : Starting cloud process 'a8230c57-6ca5-4c35-b2fb-d9cdde86966d' of type 'float []'.
INFO : Starting Cloud.Parallel<float> workflow 0e5108ef-99b1-4d0b-8332-298b753ae03b of 17 children.
INFO : Completed work item '6087c8a0-9f62-417a-a96b-351bc8a97f67' after 00:00:00.6530842
INFO : Concurrent work ite
INFO : Dequeued work item
INFO : Concurrent work ite
INFO : Completed work item
INFO : Concurrent work ite
INFO : Dequeued work item
INFO : Concurrent work ite
INFO : Completed work item
INFO : Concurrent work ite
INFO : Dequeued work item
INFO : Concurrent work ite
INFO : Completed work item
INFO : Concurrent work ite
INFO : Dequeued work item
INFO : Concurrent work ite
INFO : Completed work item
INFO : Concurrent work ite
INFO : Dequeued work item
INFO : Concurrent work ite
INFO : Completed work item
INFO : Concurrent work ite
INFO : Dequeued work item
INFO : Concurrent work ite
INFO : Completed work item
INFO : Concurrent work ite
```

MBrace.Thespian Slave Node [pid:16084, port:59716]

```
INFO : Process Id: 16084
INFO : Thespian initialized with listener WORK:59716
INFO : Initialized controller actor:
        actor://controller.WorkerControllerMsg
        utcp://work:59716/controller
        mailbox://controller/.
INFO : Responding to spawning process.
INFO : Subscribing worker to runtime hosted at 'mbrace://WORK:59269'.
INFO : Initializing worker store logger.
INFO : Initializing AppDomain pool evaluator.
INFO : Creating worker agent.
INFO : Subscribed to cluster with heartbeat interval 00:00:01
INFO : Starting Worker.
INFO : Dequeued work item '4a0fa06c-35cb-43a1-9b85-b5b948eddb19'.
INFO : Concurrent work item count increased to 1/20.
INFO : Downloading 'FSharp.Compiler.Interactive.Settings, Version=4.4.1.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50
a3a'
INFO : Downloading 'FSharp.Charting, Version=0.90.14.0, Culture=neutral, PublicKeyToken=null'
INFO : Downloading 'FSI-ASSEMBLY_06633297-ed7e-4626-9461-32114d953896_2, Version=0.0.0.0, Culture=neutral, PublicKeyToke
n=null'
INFO : Dequeued work item '4b6568ba-7759-41c7-9d69-5083263f8c8c'.
INFO : Concurrent work item count increased to 2/20.
INFO : Dequeued work item '89247bf6-71eb-49ed-b879-caa457ba6b75'.
INFO : Concurrent work item count increased to 3/20.
INFO : Downloading data dependency 'Double[][] sourceValues@'.
INFO : Downloading data dependency 'MBrace.Core.Cloud`1[System.Double[]] x@1-4'.
INFO : Downloading data dependency 'MBrace.Core.Cloud`1[System.Double][] x@1-5'.
INFO : Initializing Application Domain '94b2bf36-460c-4a15-a3b7-ebb7e85946ec'.
INFO : Thespian listening to WORK:59936 on AppDomain '94b2bf36-460c-4a15-a3b7-ebb7e85946ec'.
```

```
.57 MiB
31bf385
, PDB 4
INFO : Uploading 'FSharp.Compiler.Interactive.Settings, Version=4.4.1.0, Culture=neutral, PublicKeyToken=b03f5f7f1
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    [|for x in sourceValues -> cloud { return x |> Array.sum}|]
    |> Cloud.Parallel
    |> cluster.Run
    |> Array.sum

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
   sourceValues
   |> CloudFlow.OfArray
   |> CloudFlow.map (fun x -> x |> Array.sum )
   |> CloudFlow.reduce (+)
   |> cluster.Run

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    sourceValues
    |> CloudFlow.OfArray
    |> CloudFlow.map (fun x -> x |> Array.sum )
    |> CloudFlow.reduce (+)
    |> cluster.Run

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    sourceValues
    |> CloudFlow.OfArray
    |> CloudFlow.map (fun x -> x |> Array.sum )
    |> CloudFlow.reduce (+)
    |> cluster.Run

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
    sourceValues
    |> CloudFlow.OfArray
    |> CloudFlow.map (fun x -> x |> Array.sum )
    |> CloudFlow.reduce (+)
    |> cluster.Run

cluster.KillAllWorkers()
```

{m}brace

```
open MBrace.Core
...
let sourceValues = [|for x in 0 .. 16 -> [|for y in 0 .. 2000 -> rand()|]|]

let cluster = ThespianCluster.InitOnCurrentMachine(4, ...)

let r =
   sourceValues
   |> CloudFlow.OfArray
   |> CloudFlow.map (fun x -> x |> Array.sum )
   |> CloudFlow.reduce (+)
   |> cluster.Run

cluster.KillAllWorkers()
```

3 манипулирование данными

- repl & $f(\lambda)$
- Deedle
- Math.Net
- R Provider
- {m}brace

3 манипулирование данными

- repl & $f(\lambda)$
- Deedle
- Math.Net
- R Provider
- {m}brace
- Accord.Net Framework
- numl
- encog
- Hype
- ML от MS
- AForge.Net
- ...

4 интеграция

это dotnet

ИТОГ

все прекрасно?

нет

нет, но можно

f#

FSharp.Data

RProvider

{m}brace

python vs dotnet

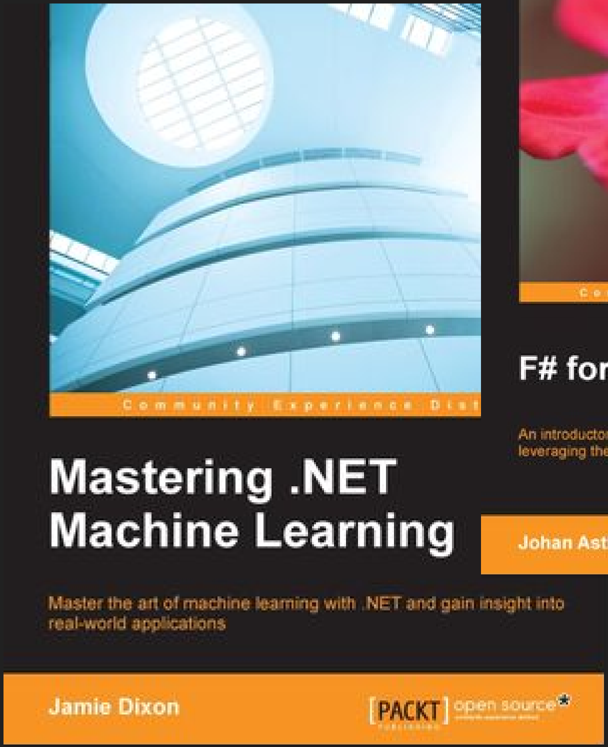
почитать

- http://fsharp.org/
- https://fslab.org/
- http://fsharp.github.io/FSharp.Data/
- http://bluemountaincapital.github.io/Deedle/
- http://bluemountaincapital.github.io/FSharpRProvider/
- https://numerics.mathdotnet.com/
- https://tahahachana.github.io/XPlot/
- https://fslab.org/FSharp.Charting/
- http://accord-framework.net
- http://numl.net/

почитать

- http://www.heatonresearch.com/encog/
- https://azure.microsoft.com/ru-ru/services/machine-learning-studio/
- http://hypelib.github.io/Hype/
- http://mbrace.io/

почитать


Community Experience Distilled

Mastering .NET Machine Learning

Master the art of machine learning with .NET and gain insight into real-world applications

Jamie Dixon

[PACKT] open source*

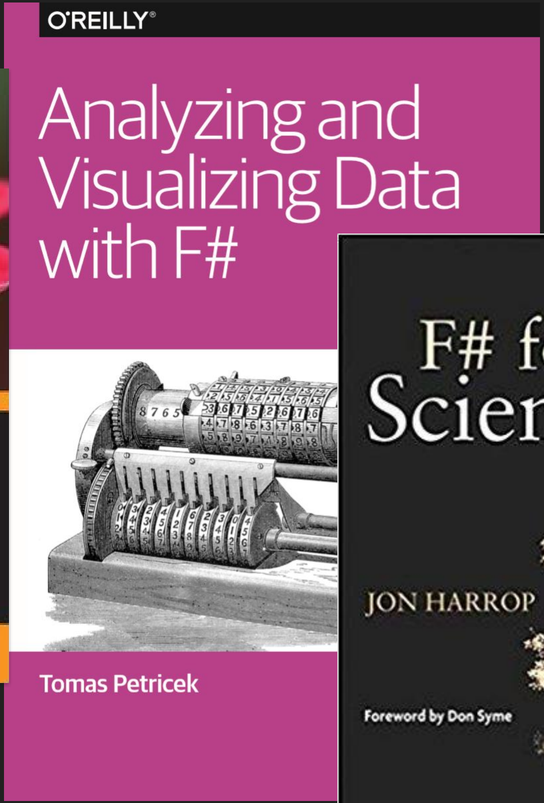
Community Experience Distilled

F# for Quantitative Finance

An introductory guide to utilizing F# for quantitative finance leveraging the .NET platform

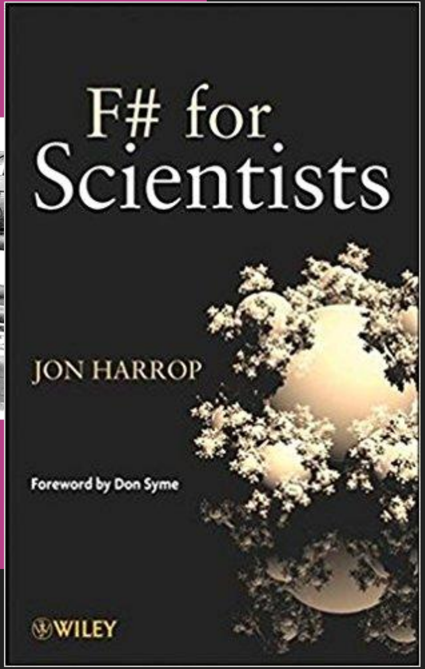
Johan Astborg

[PACKT] open source*


O'REILLY®

Analyzing and Visualizing Data with F#

Tomas Petricek


F# for Scientists

JON HARROP

Foreword by Don Syme

WILEY

вопросы?

vk.com/oxffaa
akulyakov.artem@gmail.com
github.com/0xffaa